QBUS6850

Lecture 12

Recommender Systems

❑ Topics covered

- Recommender/recommendation systems introduction
- Content based filtering
- Collaborative filtering

❑ References

- Koren, (2010): Factor in the neighbors: Scalable and accurate collaborative filtering
- http://surpriselib.com/

Learning Objectives

- Understand what is the recommender/recommendation systems
- Understand the intuition of content based filtering
- Understand the intuition of collaborative filtering
- Understand the base line approach in Koren (2010)
- Understand the kNN approach in Koren (2010)
- Understand cosine and Pearson correlation similarity function
- Be able to calculate the rating predictions
- Understand how to check the performance of recommender system

Recommendation/ recommender systems

Introduction

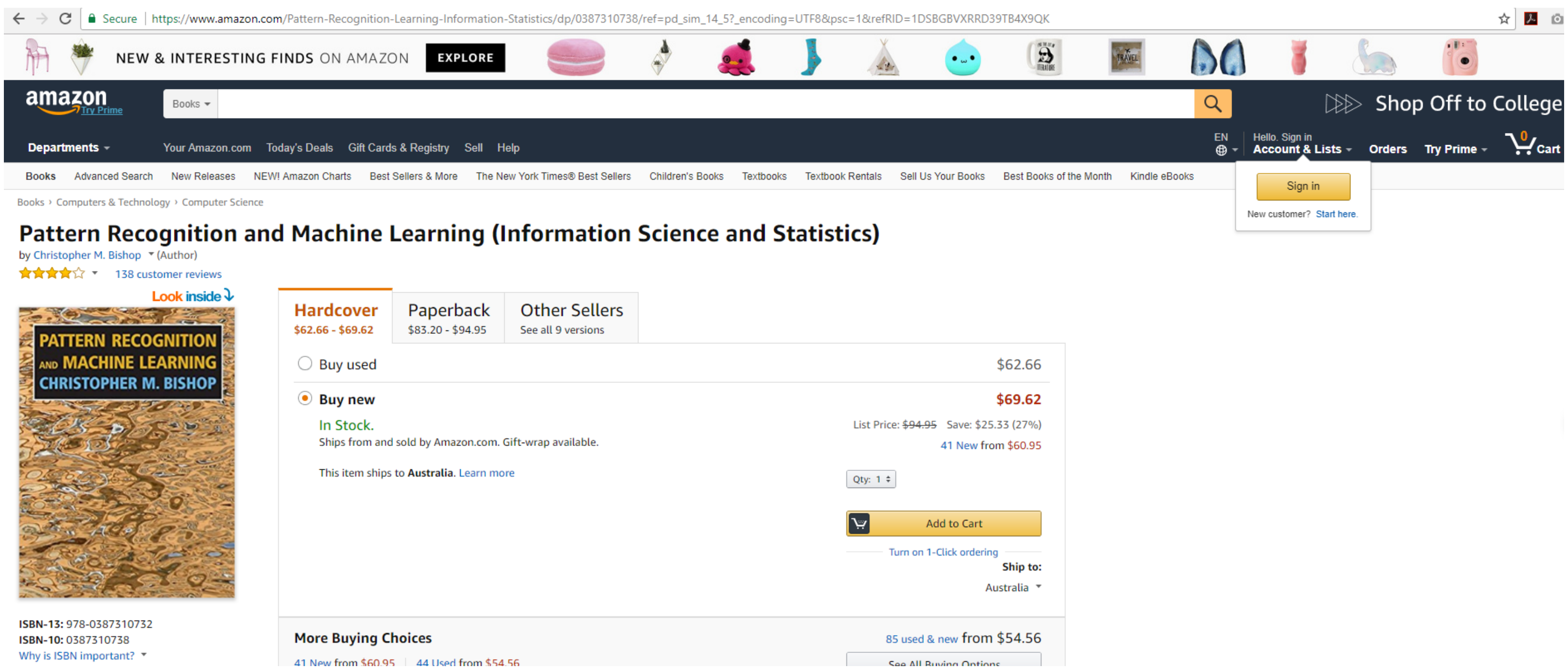
- A recommender system or a recommendation system (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item.

- Recommender systems typically produce a list of recommendations in one of two ways – through collaborative and content-based filtering or the personality-based approach.

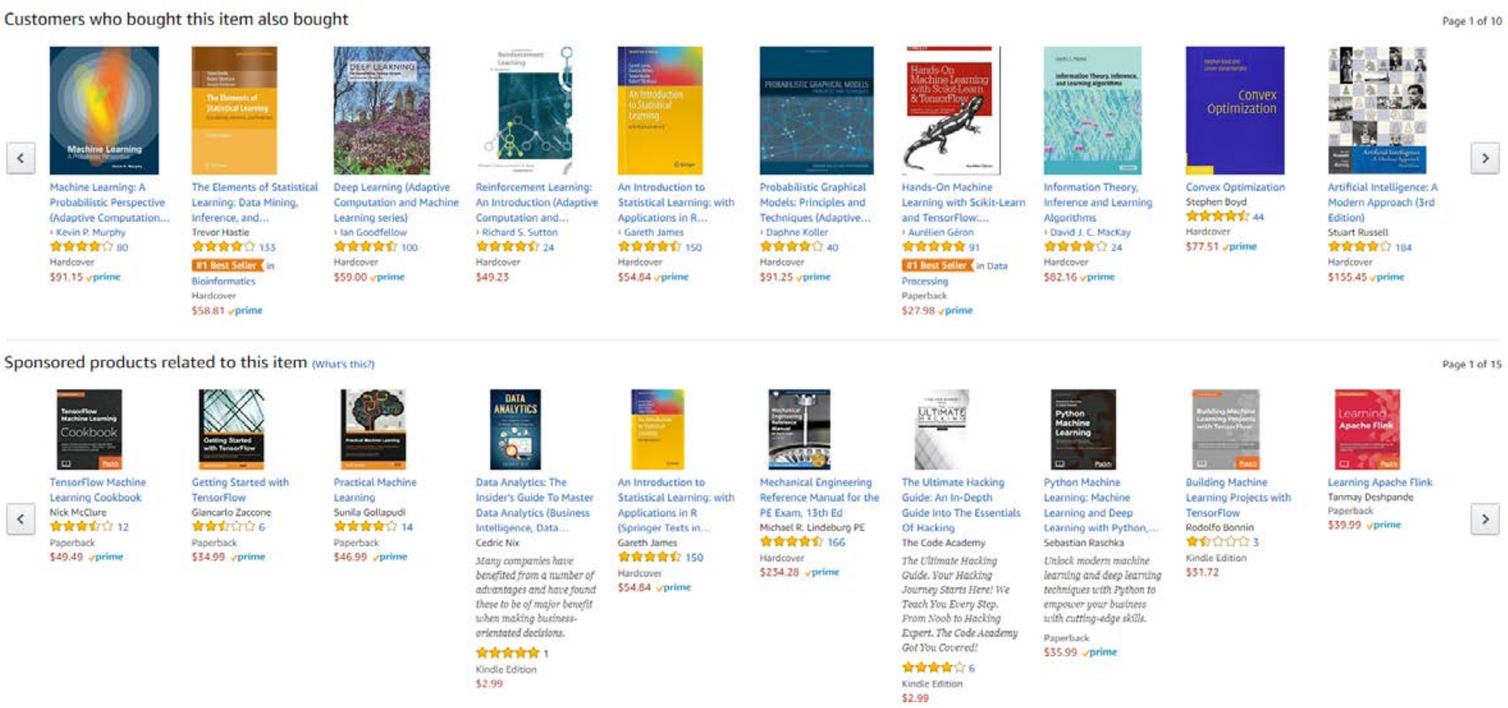
https://en.wikipedia.org/wiki/Recommender_system

THE UNIVERSITY OF SYDNEY

Customer choice



Recommendations



Applications

- Movies,
- Music,
- News,
- Books,
- Research articles,
- Search queries,
- Social tags,
- Products
- Restaurants,
- Financial services,
- Insurance,
- …..

Datasets

Movies Recommendation:
•*MovieLens* - Movie Recommendation Data Sets http://www.grouplens.org/node/73
•*Yahoo!* - Movie, Music, and Images Ratings Data Sets http://webscope.sandbox.yahoo.com/catalog.php?datatype=r
•*Jester* - Movie Ratings Data Sets (Collaborative Filtering Dataset) http://www.ieor.berkeley.edu/~goldberg/jester-data/
•*Cornell University* - Movie-review data for use in sentiment-analysis experiments http://www.cs.cornell.edu/people/pabo/movie-review-data/
Music Recommendation:
•*Last.fm* - Music Recommendation Data Sets http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/index.html
•*Yahoo!* - Movie, Music, and Images Ratings Data Sets http://webscope.sandbox.yahoo.com/catalog.php?datatype=r
•*Audioscrobbler* - Music Recommendation Data Sets http://www-etud.iro.umontreal.ca/~bergstrj/audioscrobbler_data.html
•*Amazon* - Audio CD recommendations http://131.193.40.52/data/
Books Recommendation:
•*Institut für Informatik, Universität Freiburg* - Book Ratings Data Sets http://www.informatik.uni-freiburg.de/~cziegler/BX/

https://gist.github.com/entaroadun/1653794

Content based recommendation

- **Content based recommendation:**
- Idea: if you like an item then you will also like a “similar” item, based on a set of features (**CONTENT**) of the items
- It generally works well when its easy to determine the context/properties of each item. For instance when we are recommending the same kind of item like a movie recommendation or song recommendation

https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/

Intuition

- Features for the different types of music are available
- Features capture the content of these songs, e.g. pop. Jazz, rock
- Using features of a content of the movies to make our predictions.

1: yes. 0: No

Item	Jazz or not- x1	Pop or not- x2	Rock or not- x3	Female singer or not- x4	After 2010 or not- x5
1- Take five	1	0	0	0	0
2- Strange Fruit	1	0	0	1	0
3- Call Me Maybe	**0**	**1**	**0**	**1**	**1**
4- Shake It Off	**0**	**1**	**0**	**1**	**1**
5- The Lazy Song	0	1	0	0	1
6- Sweet emotion	0	0	1	0	0
7- Still Loving You	0	0	1	0	0

Item 3 and 4 are quite similar (same) based on the listed 5 features. Therefore, if one customer listened “Call Me Maybe”, the recommendation system would probably recommend “Shake It Off”

Advantageous

Advantageous:

- Do not need data of other users
- Able to recommend to users with unique taste
- There is no “first-rater” or “cold start” problem: capable of recommending new & popular items
- Explainability: we have clear reasons of why recommend such items, as content/features lead to the recommendation

Disadvantageous

Disadvantageous:

- It is hard to find the appropriate features, e.g. how pop each song is and how rock each song is, one movie can have multiple features/content, etc
- Unable to incorporate the judgements from other users
- Never recommends items outside user's profile, while customers might have multiple interests as below example

Collaborative Filtering (CF)

Notations

- R : the set of all ratings.
- R_{train}, R_{test} and $\hat{R}$ denote the training set, the test set, and the set of predicted ratings.
- U : the set of all users. u and v denotes users.
- I : the set of all items. i and j denotes items.
- U_i : the set of all users that have rated item i.
- U_{ij} : the set of all users that have rated both items i and j.
- I_u : the set of all items rated by user u.
- I_{uv} : the set of all items rated by both users u and v.
- r_{ui} : the *true* rating of user u for item i.
- $\hat{r}_{ui}$: the *estimated* rating of user u for item i.
- b_{ui} : the baseline rating of user u for item i.
- μ : the mean of all ratings.
- μ_u : the mean of all ratings given by user u.
- μ_i : the mean of all ratings given to item i.
- $N_i^k(u)$: the k nearest neighbors of user u that have rated item i. This set is computed using a `similarity metric`.
- $N_u^k(i)$: the k nearest neighbors of item i that are rated by user u. This set is computed using a `similarity metric`.

http://surprise.readthedocs.io/en/stable/notation_standards.html#notation-standards

Comparison

In the content based recommendation:

- Features are KNOWN
- It can be very difficult and time consuming and expensive to actually try to get someone to listen each song and tell you how pop each song and how rock is each song

Collaborative filtering (Goldberg et al., 1992) is a common technique used by recommender systems:

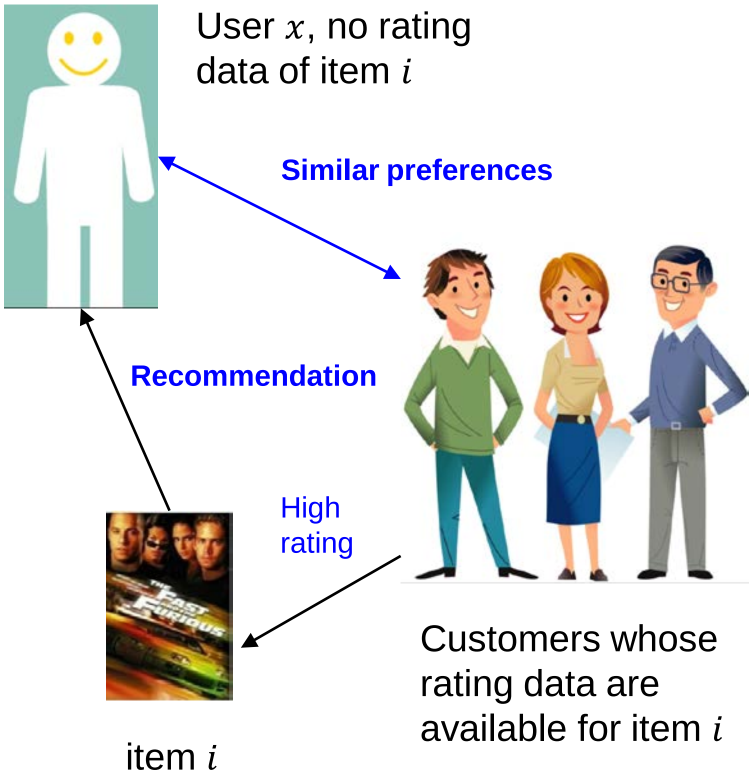
- Features are UNKNOWN
- Relies only on past user behaviour
- Is based on customers' previous transactions or product ratings and do not require the creation of explicit profiles/features
- Or we're given the data set of movies and of how the users rated them, but we have no idea how pop each song is and how rock each song is

Intuition

- Collaborating:
  - Collecting preferences or feedback information from many users
- Filtering:
  - Making automatic predictions of user ratings
- Principle: A person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

Intuition

- User x, no rating data of item i
- Find set of N other users whose ratings/taste/preferences are "**similar**" to user x
- Predict the rating of user x on item i based on ratings of N other users
- If the predicted rating of user x on item i is high, the recommend item i



Intuition

- Idea (again): If a person A likes item 1, 2, 3 and B like 2, 3, 4 then they have similar interests and A should like item 4 and B should like item 1.
- This algorithm is entirely based on the past behavior and not on the context. This makes it one of the most commonly used algorithm as it is not dependent on any additional information.
- CF attracted much of attention in the past decade, resulting in significant progress and being adopted by some successful commercial systems, including Amazon (Linden et al., 2003), TiVo (Ali and van Stam, 2004) and Netflix.
- CF systems need to compare fundamentally different objects: **items against users**. There are two primary approaches to facilitate such a comparison, which constitute the two main disciplines of CF: ***the neighborhood approach*** *and latent factor models*

Types of CF

Further, there are several types of collaborative filtering algorithms :

- **User-User** Collaborative filtering: Here we find look alike users (based on similarity) and offer products which first user's look alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every customer pair information which takes time. Therefore, for big base platforms, this algorithm is hard to implement without a very strong parallelizable system.

- **Item-Item** Collaborative filtering: It is quite similar to previous algorithm, but instead of finding customer look alike, we try finding item look alike. Once we have item look alike matrix, we can easily recommend alike items to user who have purchased any item from the store. This algorithm is far **less resource consuming** than user-user collaborative filtering. Hence, for a new user the algorithm takes far lesser time than user-user collaborate as we don't need all similarity scores between customers. And with fixed number of products, product-product look alike matrix is fixed over time.

- Other simpler algorithms: There are other approaches like market basket analysis, which generally do not have high predictive power than the algorithms described above.

https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/

The Netflix Prize

From 2006 to 2009, Netflix sponsored a competition, offering a grand prize of $1,000,000 to the team that could take an offered dataset of over 100 million movie ratings and return recommendations that were **10% more** accurate than those offered by the company's existing recommender system.

How to evaluate the accuracy?

Problem Specification

We are given ratings about *m* users and *n* items. Koren (2010) used special indexing letters for distinguishing users from items:

- **for users *u*, *v*,** and **for items *i*, *j***

A rating r_{ui} indicates the preference by user *u* of item *i*, where high values mean stronger preference. For example, values can be integers ranging from 1 (star) indicating no interest to 5 (stars) indicating a strong interest.

	User 1	User 2	User 3	User 4
Item 1	4	5		2
Item 2		1		
Item 3		4		
Item 4	5	3	2	3
Item 5	1		4	
Item 6		4	5	3

Yellow represents missing ratings

$$r_{ui} = r_{21} = ?$$

- Koren (2010) distinguished predicted ratings from known ones, by using the notation $\widehat{r_{u,i}}$ for the predicted value of $r_{u,i}$.
- Usually the vast majority of ratings are **unknown**. For example, in the Netflix data 99% of the possible ratings are missing because a user typically rates only a small portion of the movies.
- The (u, i) pairs for which $r_{u,i}$ is known are stored in the set $K = \{(u, i) | r_{u,i} \text{ is known}\}$
- In order to combat overfitting the sparse rating data, models are **regularized** so estimates are shrunk towards baseline defaults. Regularization is controlled by constants that are denoted as: λ_1, λ_2 ...
- Values of these regularization constants are determined by cross validation

THE UNIVERSITY OF SYDNEY

Baseline Estimates Approach

A baseline estimate for an unknown rating r_{ui} is denoted by b_{ui} and accounts for the user and item effects:

Global Item

$$b_{ui} = \mu + b_u + b_i$$

User

- Typical CF data exhibit large user and item effects
- Systematic tendencies for some users to give lower ratings than others
- Systematic tendencies for some items to receive higher ratings than others.
- The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average.
  - Offset per user: b_u indicates the observed deviations of user u
  - Offset per item: b_i indicates the observed deviations of item i
  - Global bias μ

THE UNIVERSITY OF SYDNEY

Example

$$b_{ui} = \overset{\text{Global}}{\overset{\downarrow}{\mu}} + \underset{\text{User}}{\underset{\uparrow}{b_u}} + \overset{\text{Item}}{\overset{\downarrow}{b_i}}$$

- For example, suppose that we want a baseline estimate for the rating of the movie Titanic by user Joe.
- Say that the average rating over all movies, $\mu = 3.7$ stars.
- Joe is a critical user, who tends to rate $b_u = -0.3$ stars lower than the average.
- Titanic is better than an average movie, so it tends to be rated $b_i = +0.5$ stars above the average
- Thus, the baseline estimate for Titanic’s rating by Joe would be 3.9 stars by calculating $b_{ui} = \mu + b_u + b_i =$ 3.7−0.3+0.5=3.9.

Estimation

Least Squares (LS) Problem

Given ratings

$$\boxed{\min_{b_u, b_i} \sum_{(u,i)\in K} \left(\boxed{r_{ui}} - \mu - b_u - b_i\right)^2} + \boxed{\lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right)}$$

Try to find b_u and b_i that fit the given rating

Avoids overfitting by penalizing the magnitudes of the parameters

Neighborhood Approach

We focus on the neighbourhood approach in our unit

- The most common approach to CF is based on neighborhood models
- Relative simplicity and intuitiveness.
- Naturally provide intuitive explanations of the reasoning behind recommendations, which often enhance user experience beyond what improved accuracy might achieve.
- Able to immediately provide recommendations based on just entered user feedback.

Neighborhood Approach

- Our goal is to predict r_{ui}: the unobserved rating by user u for item i.
- Using the similarity measure, we identify the k users (neighbours), which are most similar to user u. This set of k neighbours is denoted by $N_i^k(u)$. The predicted value of r_{ui} is calculated as:

1. A simple average of k neighbours
2. A **weighted average** based on similarity of the ratings of k neighbours
3. Yehuda, (2010) adjusted for user and item effects through the baseline estimates (not in our lecture)

The k nearest neighbors of user u that have rated item i. This set is computed based on a similarity function.

User to user

1. $$\hat{r}_{ui} = \frac{\sum_{v \in \boxed{N_i^k(u)}} r_{vi}}{k}$$

2. $$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) r_{vi}}{\sum_{v \in N_i^k(u)} \boxed{\text{sim}(u,v)}}$$

How to find k nearest neighbors?

User to user similarity

The above two predictions are based on user to user approach, we can of course use the item to item approach as in the below link:
https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBasic

Item to item

k nearest neighbors of item i that are rated by user u .

1. $$\hat{r}_{ui} = \frac{\sum_{\boxed{j \in N_u^k(i)}} r_{uj}}{k}$$

2. $$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) r_{uj}}{\sum_{j \in N_u^k(i)} \boxed{\text{sim}(i,j)}}$$

Item to item similarity

Read the formula in detail and see the difference between user to user and item to item approaches

Similarity Measurements

1. Jaccard Similarity:

- Similarity for items A and B is based on the number of users which have rated item A and B divided by the number of users who have rated either A or B; and similarity for users u and v similarly
- It is typically used where we don't have a numeric rating but just a boolean value like a product being bought or an add being clicked

2. Cosine Similarity:

- Similarity is the cosine of the angle between the 2 vectors of the item vectors of A and B
- Closer the vectors, smaller will be the angle and larger the cosine

3. Pearson Similarity

- Similarity is the pearson coefficient between the two vectors.

Jaccard Similarity

Jaccard Similarity between user *u*, *v is:*

$$\text{Jaccard} - \text{sim}(u,v) = \frac{|r_u \cap r_v|}{|r_u \cup r_v|}$$

- The number of items rated by users u and v divided by the number of items rated either user u or user v
- This can be easily extended to item-item
- Note in the “surprise” this is not used

Jaccard Similarity Example

- For users 1 and 3, they rated two movies in common, but they appear to have almost diametrically opposite opinions of these movies. A good distance measure would make them rather far apart.
- Jaccard distance ignores values in the matrix and focus only on the sets of items rated. When the matrix contains detailed ratings, the Jaccard distance loses important information.

	User 1	User 2	User 3	User 4
Item 1	4	5		2
Item 2		1		
Item 3		4		
Item 4	5	3	2	3
Item 5	1		4	
Item 6		4	5	3

User 1 and 2 have an intersection of size 2 and a union of size 6, Jaccard-sim= 2/6

In comparison, user 1 and 3 have an intersection of size 2 and a union of size 4, Jaccard-sim= 2/4

Conclusion with Jaccard-simimarity:
User 1 appears closer to 3 than to 2, which seems intuitively wrong. User 1 and 3 disagree on the two movies they both watched, while 1 and 2 seem both to have liked the two movies they watched in common.

Cosine Similarity Intuition

The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = \| \mathbf{a} \| \cdot \| \mathbf{b} \| \cos(\theta)$$

- If two vectors are similar to each other, then the **angle** between the two vectors should be small.
- The cosine similarity is just the **cosine of the angle** between two vectors
- cosine similarity is higher, the two vector are more similar

$$\text{cos-sim}(\mathbf{a}, \mathbf{b}) = \cos(\theta) = \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} = \frac{\sum_{i=1}^{d} a_i b_i}{\sqrt{\sum_{i=1}^{d} a_i^2} \sqrt{\sum_{i=1}^{d} b_i^2}}$$

Cosine Similarity Intuition

For example, the angle θ between vector $\boldsymbol{a} = [1, 1]^T$, $\boldsymbol{b} = [2, 2]^T$ is 0°, $\cos(0°) = 1$.

$$\cos - \mathrm{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\| \mathbf{a} \| \cdot \| \mathbf{b} \|} = \frac{1 \times 2 + 1 \times 2}{\sqrt{1^2 + 1^2} \sqrt{2^2 + 2^2}} = \frac{4}{\sqrt{16}} = 1$$

The cosine of 0° is 1, and cosine is less than 1 for any other angle. It is thus a **judgment of orientation and not magnitude**: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0.

The cosine similarity between user $\mathbf{u} = [1, 1]^T$, and user $\mathbf{v} = [5, 5]^T$ is also one, while this seems wrong since two user have quite different ratings

Cosine Similarity

- Compute the cosine similarity between all pairs of users (or items).
- Only **common** users (or items) are taken into account in "Surprise".
- The cosine similarity is defined as:

Users u and v both rated item i. Then sum over all such i.

User to user

$$\cos-\text{sim}(u,v)=\frac{\boxed{\sum_{i\in I_{uv}}} r_{ui}\cdot r_{vi}}{\sqrt{\sum_{i\in I_{uv}} r_{ui}^2}\sqrt{\sum_{i\in I_{uv}} r_{vi}^2}}$$

Items i and j that were both rated by user u. Then sum over all such u.

Item to item

$$\cos-\text{sim}(i,j)=\frac{\boxed{\sum_{u\in U_{ij}}} r_{ui}\cdot r_{uj}}{\sqrt{\sum_{u\in U_{ij}} r_{ui}^2}\sqrt{\sum_{u\in U_{ij}} r_{uj}^2}}$$

Cosine Similarity Example

	User 1	User 2	User 3	User 4
Item 1	4	5		
Item 2		5		2
Item 3		4		
Item 4	5	3	2	
Item 5	1		4	
Item 6			5	
Item 6				1
User mean	3.333	4.250	3.667	1.500

The cosine similarity between **user 1 and 2** is

$$\cos - \text{sim}(1,2) = \frac{4 \times 5 + 5 \times 3}{\sqrt{4^2 + 5^2}\sqrt{5^2 + 3^2}} = 0.937$$

The cosine similarity between **user 1 and 3** is

$$\cos - \text{sim}(1,3) = \frac{5 \times 2 + 1 \times 4}{\sqrt{5^2 + 1^2}\sqrt{2^2 + 4^2}} = 0.614$$

User 1 appears much closer to 2 than to 3, which a better conclusion than Jaccard similarity.

Pearson Correlation Coefficient

- If we normalize ratings, by subtracting from each rating the average rating of that user (**mean-centered**), we turn low ratings into negative numbers and high ratings into positive numbers.
- If we then take the **cosine** distance, we find that users with opposite views of the movies they viewed in common will have vectors in almost opposite directions, and can be considered as far apart as possible.
- However, users with similar opinions about the movies rated in common will have a relatively small angle between them.

	User 1	User 2	User 3	User 4
Item 1	4	5		2
Item 2		1		
Item 3		4		
Item 4	5	3	2	3
Item 5	1		4	
Item 6		4	5	3
User Mean	**3.333**	**3.400**	**3.667**	**2.667**

Mean-centered →

	User 1	User 2	User 3	User 4
Item 1	0.667	1.600		- 0.667
Item 2		- 2.400		
Item 3		0.600		
Item 4	1.667	- 0.400	- 1.667	0.333
Item 5	- 2.333		0.333	
Item 6		0.600	1.333	0.333

Pearson Correlation Coefficient

- The **mean-centered cosine similarity** is also called Pearson correlation coefficient
- Compute the Pearson correlation coefficient between all pairs of users (or items).
- Only **common** users (or items) are taken into account.
- The Pearson correlation coefficient is defined as:

User to user

Users u average

$$\text{pearson}-\text{sim}(u,v)=\frac{\sum_{i\in I_{uv}}(r_{ui}-\boxed{\mu_u})\cdot(r_{vi}-\mu_v)}{\sqrt{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)^2}\sqrt{\sum_{i\in I_{uv}}(r_{vi}-\mu_v)^2}}$$

Item to item

Items i average

$$\text{pearson}-\text{sim}(i,j)=\frac{\sum_{u\in U_{ij}}(r_{ui}-\boxed{\mu_i})\cdot(r_{uj}-\mu_j)}{\sqrt{\sum_{u\in U_{ij}}(r_{ui}-\mu_i)^2}\sqrt{\sum_{u\in U_{ij}}(r_{uj}-\mu_j)^2}}$$

Pearson Similarity Example

	User 1	User 2	User 3	User 4
Item 1	0.667	1.600		- 0.667
Item 2		- 2.400		
Item 3		0.600		
Item 4	1.667	- 0.400	- 1.667	0.333
Item 5	- 2.333		0.333	
Item 6		0.600	1.333	0.333

- The pearson similarity between **user 1 and 2** is 0.135

$$\mathrm{pearson-sim}(1,2)$$

$$= \frac{0.667 \times 1.6 + 1.667 \times (-0.4)}{\sqrt{0.667^2 + 1.667^2}\sqrt{1.6^2 + (-0.4)^2}} = 0.135$$

THE UNIVERSITY OF SYDNEY

Pearson Similarity Example

	User 1	User 2	User 3	User 4
Item 1	0.667	0.750		
Item 2		0.750		0.5
Item 3		- 0.250		
Item 4	1.667	- 1.250	- 1.667	
Item 5	- 2.333		0.333	
Item 6			1.333	
Item 6				-0.5

- The Pearson similarity between **user 1 and 3** is **-0.729**
- Given that user1 and 3 disagree on the two movies they rated in common
- So user 1 and 3 are quite further apart

$$\text{pearson}-\text{sim}(1,3)$$

$$=\frac{1.667\times(-1.667)+(-2.333)\times 0.333}{\sqrt{1.667^2+(-2.333)^2}\sqrt{(-1.667)^2+0.333^2}}=-0.729$$

Pearson Similarity Example

	User 1	User 2	User 3	User 4
Item 1	0.667	1.600		- 0.667
Item 2		- 2.400		
Item 3		0.600		
Item 4	1.667	- 0.400	- 1.667	0.333
Item 5	- 2.333		0.333	
Item 6		0.600	1.333	0.333

- The Pearson similarity between **user 1 and 4** is 0.082

$$\text{pearson}-\text{sim}(1,4) = \frac{0.667\times(-0.667)+1.667\times 0.333}{\sqrt{0.667^2+1.667^2}\sqrt{(-0.667)^2+0.333^2}} = 0.082$$

Rating Prediction

	User 1	User 2	User 3	User 4
Item 1	4	5		2
Item 2		1		
Item 3		4		
Item 4	5	3	2	3
Item 5	1		4	
Item 6	???	4	5	3
User Mean	3.333	3.400	3.667	2.667

User to user similarity:

Sim(1,2)= 0.135

Sim(1,3)= -0.729

Sim(1,4)= 0.082

Suppose k=2 in kNN based on Pearson similarity;

$u = 1;\ i = \ 6;$

$\hat{r}_{ui}$: prediction of rating of user 1 for item 6

Two neighbour users are user 2 and user 4

1. $$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} r_{vi}}{k} = \frac{4+3}{2} = 3.5$$

2. $$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \mathrm{sim}(u,v) r_{vi}}{\sum_{v \in N_i^k(u)} \mathrm{sim}(u,v)} = \frac{0.135 \times 4 + 0.082 \times 3}{0.135 + 0.082} = 3.622$$

Have a try to calculate $\hat{r}_{ui}$ based on item to item similarity.

User-user & Item-item

- In theory, user to user and item to item are dual approaches
- However, in practice, item to item outperforms user to user in many use cases
- Items are "simpler" than users:
  - Normally we have much more users than items
  - Items have a much smaller set of "genres", users have much wider tastes and preferences
  - User to user approach requires to compute every user pair information which takes time
  - Item similarity is more meaningful/popular than user similarity

Latent Factor Model

- For example: Singular value decomposition (SVD), Matrix Factorization (MF)

$$\min_{\mathbf{W},\mathbf{H}} \|\mathbf{R} - \Omega(\mathbf{WH})\|_F^2 + \lambda(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)$$

- Comprise an alternative approach by transforming both items and users to the same latent factor space, thus making them directly comparable. The latent space tries to explain ratings by characterizing both items and users on factors automatically inferred from user feedback.
- For example, when the items are movies, factors might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well defined dimensions such as depth of character development or quirkiness; or completely uninterpretable dimensions. For users, each factor measures how much the user likes movies that score high on the corresponding movie factor.

- The recent paper on this topic: https://arxiv.org/pdf/1711.10816.pdf

CF advantageous:

- CF techniques do not require domain knowledge and avoid the need for extensive data collection
- Relying directly on user behavior allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes
- CF engines work best when the user space is large (since that is their source of data). Content based engines are more or less insensitive to user size.

CF disadvantageous:

- CF suffers from the "new item" problem much more than CB engines

THE UNIVERSITY OF SYDNEY

Hybrid Recommender Systems

The Problem

Algorithm selects item *j* with item features $\boldsymbol{x}_j$

(keywords, content categories, ...)

User u visits with user features $\mathbf{x}_u$

(demographics, browse history, geo-location, search history, …)

(u,j): response r_{uj}

(click/no-click)

Which item should we select?

- The one with highest predicted CTR **Exploit**
- The one most useful for improving the CTR prediction model **Explore**

THE UNIVERSITY OF SYDNEY

CB+CF

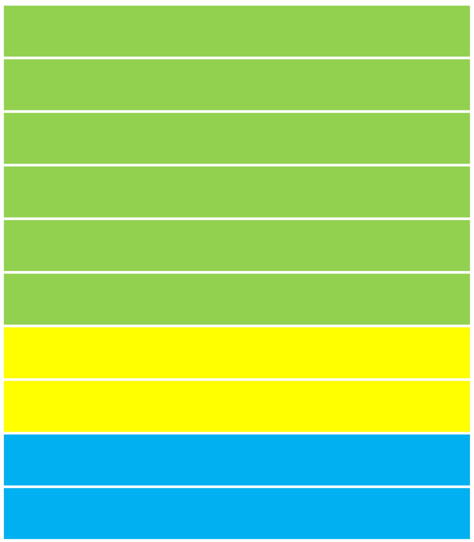
- **Hybrid approach**: combining content based filtering and collaborative filtering could be more effective in some cases.
- Hybrid approaches can be implemented in several ways:
  - by making content-based and collaborative-based predictions separately and then combining them;
  - by adding content-based capabilities to a collaborative-based approach (and vice versa);
  - by unifying the approaches into one model (see (Adomavicius and Tuzhilin, 2005) for a complete review of recommender systems).
- Several studies empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem.

https://en.wikipedia.org/wiki/Recommender_system

- Netflix is a good example of the use of hybrid recommender systems.
- The website makes recommendations by:
  - Comparing the watching and searching habits of similar users (collaborative filtering)
  - As well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering).

Evaluating the Prediction Performance

Training, validation and test sets



Training set: Estimate the model

We can still use cross validation to select the best model and avoid overfitting

Validation set: select the model

Test set: Estimate the prediction/generalization error

Prediction Evaludation

Compare the predictions against withheld ratings (test set)

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{\sum_{\hat{r}_{ui} \in \hat{R}} | r_{ui} - \hat{r}_{ui} |}{|\hat{R}|}$$

Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{\sum_{\hat{r}_{ui} \in \hat{R}} \left(r_{ui} - \hat{r}_{ui} \right)^2}{|\hat{R}|}}$$

$\hat{R}$ denotes the set of predicted ratings.

Low-Rank Matrix Completion (optional)

- Many users of Netflix have similar or shared tastes. Their individual taste can be described as a combination of many other peoples tastes.
- Mathematically this means that each users rating vector (their taste) is a linear combination of other users.
- Therefore, the total rating matrix must be relatively low-rank, so we should try to find the lowest rank matrix while keeping the existing user ratings fixed".

Problem formulation, objective function and intuition can be found in:

- Section 1 of "Numerical algorithms for low-rank matrix completion problems, (Michenková, 2011)"
- Section 1 of "Low-Rank Matrix Completion, (Kennedy, 2013)"